As a matter of fact, the main difference lies within the directives that can be employed within either block—some can be inserted in an `if` block and some can't; on the contrary, almost all the directives are authorized within a `location` block, as you probably noticed in the directive listings so far. In general, it's best to only insert the directives from the Rewrite module within an `if` block, as other directives were not originally intended for such usage.

Directives

The Rewrite module provides you with a set of directives that do more than just rewriting a URI. The following table describes these directives, along with the context in which they can be employed:

Directive	Description
`rewrite` Context: `server`, `location`, `if`	As discussed previously, the `rewrite` directive allows you to rewrite the URI of the current request, thus resetting the treatment of the said request. Syntax: `rewrite regexp replacement [flag];` Where `regexp` is the regular expression that the URI should match in order for the replacement to apply. Flag may take one of the following values: • `last`: The current rewrite rule should be the last to be applied. After its application, the new URI is processed by Nginx, and a `location` block is searched for. However, further rewrite instructions will be disregarded. • `break`: The current rewrite rule is applied, but Nginx does not initiate a new request for the modified URI (does not restart the search for matching `location` blocks). All further rewrite directives are ignored. • `redirect`: Returns a `302 Moved temporarily` HTTP response, with the replacement URI set as the value of the `location` header. • `permanent`: Returns a `301 Moved permanently` HTTP response, with the replacement URI set as the value of the `location` header. • If you specify a URI beginning with `http://` as the replacement URI, Nginx will automatically use the `redirect` flag. • Note that the request URI processed by the directive is a relative URI: It does not contain the hostname and protocol. For a request such as `http://website.com/documents/page.html`, the request URI is `/documents/page.html`.

Directive	Description

	• Is decoded: The URI corresponding to a request such as `http://website.com/my%20page.html` would be `/my page.html` (in the encoded URI, %20 indicates a white space character).
	• Does not contain arguments: For a request such as `http://website.com/page.php?id=1&p=2`, the URI would be `/page.php`. When rewriting the URI, you don't need to consider including the arguments in the replacement URI—Nginx does it for you. If you want Nginx not to include the arguments after the rewritten URI, you must insert a `?` character at the end of the replacement URI: `rewrite ^/search/(.*)$ /search.php?q=$1?`.
	• Examples:
	`rewrite ^/search/(.*)$ /search.php?q=$1;`
	`rewrite ^/search/(.*)$ /search.php?q=$1?;`
	`rewrite ^ http://website.com;`
	`rewrite ^ http://website.com permanent;`

`break`
Context: `server`, `location`, `if`
	The `break` directive is used to prevent further rewrite directives. Past this point, the URI is fixed and cannot be altered.
	Example:
	`if (-f $uri) {`
	`  break; # break if the file exists`
	`}`
	`if ($uri ~ ^/search/(.*)$) {`
	`  set $query $1;`
	`  rewrite ^ /search.php?q=$query?;`
	`}`
	This example rewrites `/search/anything`-like queries to `/search.php?q=anything`. However, if the requested file exists (such as `/search/index.html`), the break instruction prevents Nginx from rewriting the URI.

Directive	Description

`return`
Context: `server`, `location`, `if`
	Interrupts the processing of the request, and returns the specified HTTP status code or specified text.
	Syntax: `return code | text;`
	Where the code is one of the following status codes: `204`, `400`, `402` to `406`, `408`, `410`, `411`, `413`, `416`, and `500` to `504`. In addition, you may use the Nginx-specific code `444` in order to return a `HTTP 200 OK` status code with no further response header or body. Alternatively, you may also specify a raw text value that will be returned to the user as the response body. This comes in handy when testing whether your request URIs fall within particular location blocks.
	Example:
	`if ($uri ~ ^/admin/) {`
	`  return 403;`
	`  # the instruction below is NOT executed`
	`  # as Nginx already completed the request`
	`  rewrite ^ http://website.com;`
	`}`

`set`
Context: `server`, `location`, `if`
	Initializes or redefines a variable. Note that some variables cannot be redefined, for example, you are not allowed to alter `$uri`.
	Syntax: `set $variable value;`
	Examples:
	`set $var1 "some text";`
	`if ($var1 ~ ^(.*) (.*)$) {`
	`  set $var2 $1$2; #concatenation`
	`  rewrite ^ http://website.com/$var2;`
	`}`

`uninitialized_variable_warn` Context: `http`, `server`, `location`, `if`	If set to `on`, Nginx will issue log messages when the configuration employs a variable that has not yet been initialized. Syntax: `on` or `off` `uninitialized_variable_warn on;`
`rewrite_log` Context: `http`, `server`, `location`, `if`	If set to `on`, Nginx will issue log messages for every operation performed by the rewrite engine at the `notice` error level (see `error_log` directive). Syntax: `on` or `off` Default value: `off` `rewrite_log off;`